Combining Usage, Content, and Structure Data to Improve Web Site Recommendation*

Jia Li and Osmar R. Zaïane

Department of Computing Science, University of Alberta
Edmonton AB, Canada
{jial, zaiane}@cs.ualberta.ca

Abstract. Web recommender systems anticipate the needs of web users and provide them with recommendations to personalize their navigation. Such systems had been expected to have a bright future, especially in e-commerce and e-learning environments. However, although they have been intensively explored in the Web Mining and Machine Learning fields, and there have been some commercialized systems, the quality of the recommendation and the user satisfaction of such systems are still not optimal. In this paper, we investigate a novel web recommender system, which combines usage data, content data, and structure data in a web site to generate user navigational models. These models are then fed back into the system to recommend users shortcuts or page resources. We also propose an evaluation mechanism to measure the quality of recommender systems. Preliminary experiments show that our system can significantly improve the quality of web site recommendation.

1 Introduction

A web recommender system is a web-based interactive software agent. A WRS attempts to predict user preferences from user data and/or user access data for the purpose of facilitating and personalizing users' experience on-line by providing them with recommendation lists of suggested items. The recommended items could be products, such as books, movies, and music CDs, or on-line resources, such as web pages or on-line activities (*Path Prediction*) [JFM97]. Generally speaking, a web recommender system is composed of two modules: an off-line module and an on-line module. The off-line module pre-processes data to generate user models, while the on-line module uses and updates the models on-the-fly to recognize user goals and predict a recommendation list.

In this paper, we investigate the design of a web recommender system to recommend on-line resources using content, structure, as well as the usage of web pages for a web site to model users and user needs. Our preliminary goals are to recommend on-line learning activities in an e-learning web site, or recommend shortcuts to users in a given web site after predicting their information needs.

One of the earliest and widely used technologies for building recommender systems is *Collaborative Filtering* (CF) [SM95] [JLH99]. CF-based recommender

* Research funded in part by the Alberta Ingenuity Funds and NSERC Canada.

systems aggregate explicit user ratings or product preferences in order to generate user profiles, which recognize users' interests. A product is recommended to the current user if it is highly rated by other users who have similar interests. The CF-based techniques suffer from several problems [SKKR00]. They rely heavily on explicit user input (e.g., previous customers' rating/ranking of products), which is either unavailable or considered intrusive. With sparsity of such user input, the recommendation precision and quality drop significantly. The second challenge is related to the system scalability and efficiency. Indeed, user profile matching has to be performed as an on-line process. For very large datasets, this may lead to unacceptable latency for providing recommendations.

In recent years there has been increasing interest in applying *web usage mining* techniques to build web recommender systems. Web usage recommender systems take web server logs as input, and make use of data mining techniques such as association rules and clustering to extract navigational patterns, which are then used to provide recommendations. Web server logs record user browsing history, which contains much hidden information regarding users and their navigation. They could, therefore, be a good alternative to the explicit user rating or feedback in deriving user models. In web usage recommender systems, navigational patterns are generally derived as an off-line process. The most commonly used approach for web usage recommender systems is using association rules to associate page hits [SCDT00] [FBH00] [LAR00] [YHW01]. We will test our approach againt this general technique.

However, a web usage recommender system which focuses solely on web server logs has its own problems:

- Incomplete or Limited Information Problem: A number of heuristic assumptions are typically made before applying any data mining algorithm; as a result, some patterns generated may not be proper or even correct.
- Incorrect Information Problem: When a web site visitor is lost, the clicks made by this visitor are recorded in the log, and may mislead future recommendations. This becomes more problematic when a web site is badly designed and more people end up visiting unsolicited pages, making them seem popular.
- Persistence Problem: When new pages are added to a web site, because they have not been visited yet, the recommender system may not recommend them, even though they could be relevant. Moreover, the more a page is recommended, the more it may be visited, thus making it look popular and boost its candidacy for future recommendation.

To address these problems, we propose an improved web usage recommender system. Our system attempts to use web server logs to model user navigational behaviour, as other web usage recommender systems do. However, our approach differs from other such systems in that we also combine textual content and connectivity information of web pages, which also do not require user input. We demonstrate that this approach improves the quality of web site recommendation. The page textual content is used to pre-process log data to model content coherent visit sub-sessions, which are then used to generate more accurate

users' navigational patterns. Structure data, i.e., links between pages, are used to expand navigational patterns with a rich relevant content. The connectivity information is also used to compute the importance of pages for the purpose of ranking recommendations.

A few hybrid web recommender systems have been proposed in the literature [MDL+00] [NM03]. [MDL+00] adopts a clustering technique to obtain both site usage and site content profiles in the off-line phase. In the on-line phase, a recommendation set is generated by matching the current active session and all usage profiles. Similarly, another recommendation set is generated by matching the current active session and all content profiles. Finally, a set of pages with the maximum recommendation value across the two recommendation sets is presented as recommendation. This is called a weighted hybridization method [Bur02]. In [NM03], the authors use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three kinds of navigational patterns in the off-line phase. In the on-line phase, recommendation sets are selected from the different navigational models, based on a localized degree of hyperlink connectivity with respect to a user's current location within the site. This is called a switching hybridization method [Bur02]. Whether using the weighted method or the switching method, the combination in these systems happens only in the on-line phase. Our approach combines usage data and content data in the off-line phase to generate content coherent navigational models, which could be a better model for users' information needs. Still in the off-line phase, we further combine structure data to improve the models. Also in the off-line process, we use hyperlinks to attach a rating to web resources. This rating is used during the on-line phase for ranking recommendations.

The contributions of this paper are as follows: First, we propose a novel web recommender system, which combines and makes full use of all three available channels: usage, content, and structure data. This combination is done off-line to improve efficiency, i.e, low latency. Second, we propose a novel users' navigational model. Rather than representing the information need as a sequence of visitation clicks in a visit, our model assumes different information needs in the same visit; third, we address all three problems mentioned above by combing all available information channels.

This paper is organized as follows: Section 2 presents the off-line module of our system, which pre-processes available usage and web site data, as well as our on-line module, which generates the recommendation list. Section 3 presents experimental results assessing the performance of our system.

2 Architecture of a Hybrid Recommender System

As most web usage recommender systems, our system is composed of two modules: an off-line component, which pre-processes data to generate users' navigational models, and an on-line component which is a real-time recommendation engine. Figure 1 depicts the general architecture of our system. Entries in a web server log are used to identify users and visit sessions, while web pages or

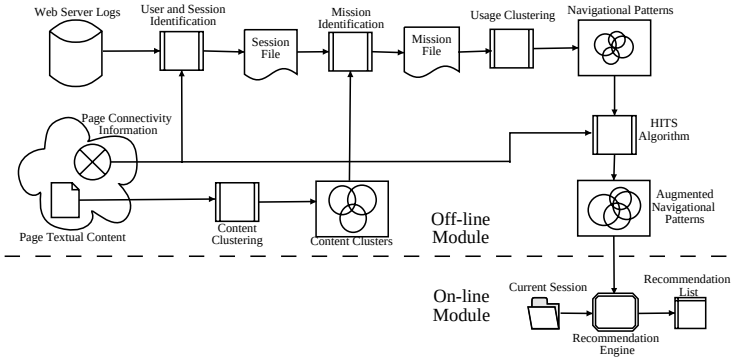


Fig. 1. System Architecture

resources in the site are clustered based on their content. These clusters are used to scrutinize the discovered web sessions in order to identify what we call *missions*. A *mission* is a sub-session with a consistent goal. These *missions* are in turn clustered to generate navigational patterns, and augmented with their linked neighbourhood and ranked based on resource connectivity, using the *hub* and *authority* idea [Kle99]. These new clusters (i.e., augmented navigational patterns) are provided to the recommendation engine. When a visitor starts a new session, the session is matched with these clusters to generate a recommendation list. The details of the whole process are given below.

2.1 User and Visit Session Identification

A web log is a text file which records information regarding users' requests to a web server. A typical web log entry contains a client address, the requested date address, a timestamp, and other related information. We use similar pre-processing techniques as in [CMS99] to identify individual users and sessions. For sessionizing, we chose an idle time of 30 minutes.

2.2 Visit Mission Identification

The last data pre-processing step proposed in [CMS99] is transaction identification, which divides individual visit sessions into transactions. In [CMS99], two transaction identification approaches are proposed: *Reference Length* approach and *Maximal Forward Reference* approach, both of which have been widely applied in web mining techniques. Rather than dividing sessions into arbitrary transactions, we identify sub-sessions with coherent information needs. We call these sub-sessions *missions*. We assume that a visitor may have different information needs to fulfill during a visit, but we make no assumption on the sequence in which these needs are fulfilled. In the case of transactions in [CMS99] it is assumed that one information need is fulfilled after the other. A *mission* would model a sub-session related to one of these information needs, and would allow

overlap between missions, which would represent a concurrent search in the site. While in the transaction-based model, pages are labeled as *content* pages and *auxiliary* pages, and a transaction is simply a sequence of auxiliary pages that ends with a content page, in our mission-based model, the identified sequence is based on the real content of pages. Indeed, a content page in the transaction-based model is identified simply based on the time spent on that page, or on backtracking in the visitor's navigation. We argue that missions could better model users' navigational behavior than transaction. In the model we propose, users visit a web site with concurrent goals, i.e., different information needs. For example, a user could fulfill two goals in a visit session: a, b, c, d, in which pages a and c contribute to one goal, while pages b and d contribute to the other. Since pages related to a given goal in a visit session are supposed to be content coherent, whether they are neighbouring each other or not, we use page content to identify missions within a visit session.

All web site pages are clustered based on their content, and these clusters are used to identify content coherent clicks in a session. Let us give an example to illustrate this point. Suppose the text clustering algorithm groups web pages *a*, *b*, *c*, and *e*, web pages *a*, *b*, *c*, and *f*, and web pages *a*, *c* and *d* into three different content clusters (please note that our text clustering algorithm is a soft clustering one, which allows a web page to be clustered into several clusters). Then for a visit session: *a*, *b*, *c*, *d*, *e*, *f*, our system identifies three missions as follows: mission 1: (a, b, c, e) ; mission 2: (a, b, c, f); and mission 3: (a, c, d). As seen in this example, mission identification in our system is different from transaction identification in that we can group web pages into one mission even if they are not sequential in a visit session. We can see that our mission-based model generates the transaction-based model, since missions could become transactions if visitors fulfill their information needs sequentially.

To cluster web pages based on their content, we use a modified version of the DC-tree algorithm [WF00]. Originally, the DC-tree algorithm was a hard clustering approach, prohibiting overlap of clusters. We modified the algorithm to allow web pages to belong to different clusters. Indeed, some web pages could cover different topics at the same time. In the algorithm, each web page is represented as a keyword vector, and organized in a tree structure called the DC-tree. The algorithm does not require the number of clusters to discover as a constraint, but allows the definition of cluster sizes. This was the appealing property which made us select the algorithm. Indeed, we do not want either too large or too small content cluster sizes. Very large clusters cannot help capture missions from sessions, while very small clusters may break potentially useful relations between pages in sessions.

2.3 Content Coherent Navigational Pattern Discovery

Navigational patterns are sets of web pages that are frequently visited together and that have related content. These patterns are used by the recommender system to recommend web pages, if they were not already visited. To discover these navigational patterns, we simply group the missions we uncovered from the

web server logs into clusters of sub-sessions having commonly visited pages. Each of the resulting clusters could be viewed as a user's navigation pattern. Note that the patterns discovered from missions possess two characteristics: usage cohesive and content coherent. Usage cohesiveness means the pages in a cluster tend to be visited together, while content coherence means pages in a cluster tend to be related to a topic or concept. This is because missions are grouped according to content information. Since each cluster is related to a topic, and each page is represented in a keyword vector, we are able to easily compute the topic vector of each cluster, in which the value of a keyword is the average of the corresponding values of all pages in the cluster. The cluster topic is widely used in our system, in both the off-line and on-line phases (see below for details). The clustering algorithm we adopted for grouping missions is *PageGather* [PE98]. This algorithm is a soft clustering approach allowing overlap of clusters. Instead of attempting to partition the entire space of items, it attempts to identify a small number of high quality clusters based on the *clique* clustering technique.

2.4 Navigational Pattern Improved with Connectivity

The missions we extracted and clustered to generate navigational patterns are primarily based on the sessions from the web server logs. These sessions exclusively represent web pages or resources that were visited. It is conceivable that there are other resources not yet visited, even though they are relevant and could be interesting to have in the recommendation list. Such resources could be, for instance, newly added web pages or pages that have links to them not evidently presented due to bad design. Thus, these pages or resources are never presented in the missions previously discovered. Since the navigational patterns, represented by the clusters of pages in the missions, are used by the recommendation engine, we need to provide an opportunity for these rarely visited or newly added pages to be included in the clusters. Otherwise, they would never be recommended. To alleviate this problem, we expand our clusters to include the connected neighbourhood of every page in a mission cluster. The neighbourhood of a page p is the set of all the pages directly linked from p and all the pages that directly link to p. Figure 3(B) illustrates the concept of neighbourhood expansion. This approach of expanding the neighbourhood is performed as follows: we consider each previously discovered navigational pattern as a set of seeds. Each seed is supplemented with pages it links to and pages from the web site that link to it. The result is what is called a connectivity graph which now represents our augmented navigational pattern. This process of obtaining the connectivity graph is similar to the process used by the HITS algorithm [Kle99] to find the *authority* and *hub* pages for a given topic. The difference is that we do not consider a given topic, but start from a mission cluster as our set of seeds. We also consider only internal links, i.e., links within the same web site. After expanding the clusters representing the navigational patterns, we also augment the keyword vectors that label the clusters. The new keyword vectors that represent the augmented navigational patterns have also the terms extracted from the content of augmented pages.

We take advantage of the built connectivity graph by cluster to apply the HITS algorithm in order to identify the *authority* and *hub* pages within a given cluster. These measures of *authority* and *hub* allow us to rank the pages within the cluster. This is important because at real time during the recommendation, it is crucial to rank recommendations, especially when the recommendation list is long. *Authority* and *hub* are mutually reinforcing [Kle99] concepts. Indeed, a good *authority* is a page pointed to by many good *hub* pages, and a good *hub* is a page that points to many good *authority* pages. Since we would like to be able to recommend pages newly added to the site, in our framework, we consider only the *hub* measure. This is because a newly added page would be unlikely to be a good authoritative page, since not many pages are linked to it. However, a good new page would probably link to many *authority* pages; it would, therefore, have the chance to be a good *hub* page. Consequently, we use the *hub* value to rank the candidate recommendation pages in the on-line module.

2.5 The Recommendation Engine

The previously described process consists of pre-processing done exclusively off-line. When a visitor starts a new session in the web site, we identify the navigation pattern after a few clicks and try to match on-the-fly with already captured navigational patterns. If they were matched, we recommend the most relevant pages in the matched cluster. Identifying the navigational pattern of the current visitor consists of recognizing the current focused topic of interest to the user. A study in [CDG$^+$98] shows that looking on either side of an anchor (i.e., text encapsulated in a *href* tag) for a window of 50 bytes would capture the topic of the linked pages. Based on this study, we consider the anchor clicked by the current user and its neighbourhood on either side as the contextual topic of interest. The captured topics are also represented by a keyword vector which is matched with the keyword vectors of the clusters representing the augmented navigational patterns. From the best match, we get the pages with the best *hub* value and provide them in a recommendation list, ranked by the *hub* values. To avoid supplying a very large list of recommendations, the number of recommendations is adjusted according to the number of links in the current page: we simply make this number proportional to the number of links in the current page. Our goal is to have a different recommendation strategy for different pages based on how many links the page already contains. Our general strategy is to give $\sqrt{n}$ best recommendations (n is the number of links), with a maximum of 10. The limit of 10 is to prevent adding noise and providing too many options. The relevance and importance of recommendations is measured with the *hub* value already computed off-line.

3 Experimental Evaluation

To evaluate our recommendation framework, we tested the approach on a generic web site. We report herein results with the web server log and web site of the

Computing Science Department of the University of Alberta, Canada. Data was collected for 8 months (Sept. 2002 – Apr. 2003), and partitioned the data into months. On average, each monthly partition contains more than 40,000 pages, resulting in on average 150,000 links between them. The log of each month averaged more than 200,000 visit sessions, which generated an average of 800,000 missions per month. The modified DC-tree content clustering algorithm generated about 1500 content clusters, which we used to identify the missions per month.

3.1 Methodology

Given the data partitioned per month as described above, we adopt the following empirical evaluation: one or more months data is used for building our models (i. e., training the recommender system), and the following month or months for evaluation. The idea is that given a session s from a month m, if the recommender system, based on data from month $m-1$ and some prefix of the session s, can recommend a set of pages p_i that contain some of the pages in the suffix of s, then the recommendation is considered accurate. Moreover, the distance in the number of clicks between the suffix of s and the recommended page p_i is considered a gain (i.e., a shortcut). More precisely, we measure the *Recommendation Accuracy* and the *Shortcut Gain* as described below.

Recommendation Accuracy is the ratio of correct recommendations among all recommendations, and the correct recommendation is the one that appears in the suffix of a session from which the prefix triggers the recommendation. As an example, consider that we have S visit sessions in the test log. For each visit session s, we take each page p and generate a recommendation list *R(p)*. *R(p)* is then compared with the remaining portion of s (i.e., the suffix of s). We denote this portion *T(p)* (T stands for Tail). The recommendation accuracy for a given session would be how often *T(p)* and *R(p)* intersect. The general formula for *recommendation accuracy* is defined as:

$$RecommendationAccuracy = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p R(p)\right|}}{S}$$

The *Shortcut Gain* measures how many clicks the recommendation allows users to save if the recommendation is followed. Suppose we have a session a, b, c, d, e, and at page b, the system recommends page e; then if we follow this advice, we would save two hops (i.e., pages c and d). There is an issue in measuring this shortcut gain when the recommendation list contains more than one page in the suffix of the session. Should we consider the shortest gain or the longest gain? To solve this problem, we opted to distinguish between *key* pages and *auxiliary* pages. A *key* page is a page that may contain relevant information and in which a user may spend some time. An *auxiliary* page is an intermediary page used for linkage and in which a user would spend a relatively short time. In our experiment, we use a threshold of 30 seconds as this distinction. Given these

two types of pages, a shortcut gain is measured as being the smallest jump gain towards a *key* page that has been recommended. If no *key* page is recommended, then it is the longest jump towards an *auxiliary* page. The set of pages in the session we go through with the assistance of the recommender system is called the shortened session *s'*. For the total S visit sessions in the test log, *Shortcut Gain* can be computed as:

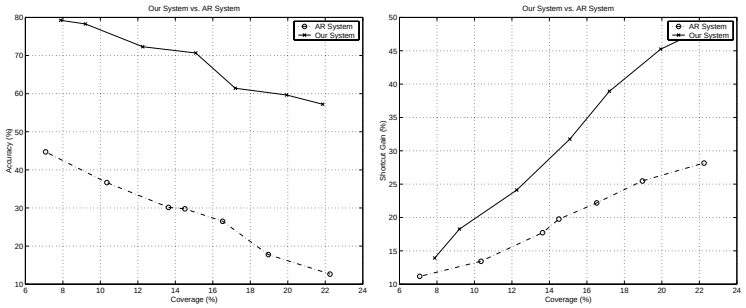
$$ShortcutGain = \frac{\sum_s \frac{|s| - |s'|}{|s|}}{S}$$

In addition, we would like to compute the *Coverage* of a recommender system, which measure the ability of a system to produce all pages that are likely to be visited by users. The concept is similar to what is called *recall* in information retrieval. *Coverage* is defined as:

$$RecommendationCoverage = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p T(p)\right|}}{S}$$

3.2 Results

Our first experiment varies the *Coverage* to see the tendency of the *Recommendation Accuracy*, as depicted in Figure 2(A). For the purpose of comparison, we also implement an Association Rule Recommender System, the most commonly used approach for web mining based recommender systems, and record its performance in the same figure. As expected, the accuracy decreases when the we increase coverage. However, our system was consistently superior to the *Association Rule* system by at least 30%.



(A) (B)

Fig. 2. Performance Comparison: our system vs. *Association Rule* Recommender System. (A): *Recommendation Accuracy* (B): *Shortcut Gain*

We next varied the *coverage* to test the *Shortcut Gain*, both with our system and with the *Association Rule* System, as illustrated in Figure 2(B).

From Figure 2(B), we can see that in the low boundary where the *Coverage* is lower than 8%, the *Shortcut Gain* of our system is close to that of the *AR* system. With the increase of the *Coverage*, however, our system can achieve an increasingly superior *Shortcut Gain* than the latter, although the performance of both systems continues to improve.

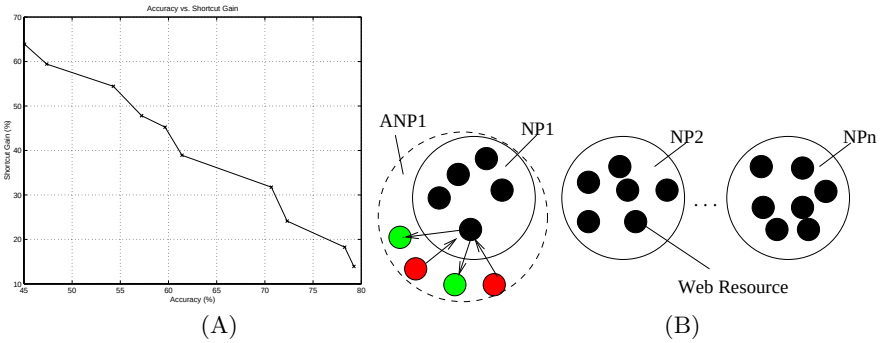


Fig. 3. (A) *Accuracy* vs. *Shortcut Gain*. (B) Navigational Patterns(NPs) and Augmented Navigational Patterns(ANPs)

Figure 3(A) depicts the relationship of *Recommendation Accuracy* and *Shortcut Gain* in our system. It shows that *Recommendation Accuracy* is inversely proportional to the *Shortcut Gain*. Our study draws the same conclusion from the *Association Rule* recommender system. We argue this is an important property of a usage-based web recommender system, and therefore, how to adjust and balance between the *Accuracy* and *Shortcut Gain* for a web recommender system to achieve the maximum benefit is a question that should be investigated. Some web sites, e.g., those with high link density, may favour a recommender system with high *Accuracy*, while some others may favor a system with high *Shortcut Gain*.

4 Conclusion

In this paper, we present a framework for a combined web recommender system, in which users' navigational patterns are automatically learned from web usage data and content data. These navigational patterns are then used to generate recommendations based on a user's current status. The items in a recommendation list are ranked according to their importance, which is in turn computed based on web structure information. Our prelimary experiments show that the combination of usage, content, and structure of data in a web recommender system has the potential to improve the quality of the system, as well as to keep the recommendation up-to-date. However, there are various ways to combine these different channels. Our future work in this area will include investigating different methods of combination.

References

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. In *User Modeling and User-Adapted Interaction*, 2002.

[CDG+98] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[FBH00] Xiaobin Fu, Jay Budzik, and Kristian J. Hammond. Mining navigation history for recommendation. In *Intelligent User Interfaces*, pages 106–112, 2000.

[JFM97] Thorsten Joachims, Dayne Freitag, and Tom M. Mitchell. Web watcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777, 1997.

[JLH99] Al Borchers John Riedl Jonathan L. Herlocker, Joseph A. Konstan. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230 – 237, 1999.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[LAR00] C. Lin, S. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining, 2000.

[MDL+00] Bamshad Mobasher, Honghua Dai, Tao Luo, Yuqing Sun, and Jiang Zhu. Integrating web usage and content mining for more effective personalization. In *EC-Web*, pages 165–176, 2000.

[NM03] Miki Nakagawa and Bamshad Mobasher. A hybrid web personalization model based on site connectivity. In *Fifth WebKDD Workshop*, pages 59–70, 2003.

[PE98] Mike Perkowitz and Oren Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *AAAI/IAAI*, pages 727–732, 1998.

[SCDT00] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[SKKR00] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[SM95] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[WF00] W. Wong and A. Fu. Incremental document clustering for web page classification, 2000.

[YHW01] Arbee L.P. Chen Yi-Hung Wu, Yong-Chuan Chen. Enabling personalized recommendation on the web based on user interests and behaviors. In *11th International Workshop on research Issues in Data Engineering*, 2001.